# Formal Analysis of Security Protocols

Shruti Biswal

*COMS/AERE 507X*

**Abstract**

Suspendisse potenti. Suspendisse quis sem elit, et mattis nisl. Phasellus consequat erat eu velit rhoncus non pharetra neque auctor. Phasellus eu lacus quam. Ut ipsum dolor, euismod aliquam congue sed, lobortis et orci. Mauris eget velit id arcu ultricies auctor in eget dolor. Pellentesque suscipit adipiscing sem, imperdiet laoreet dolor elementum ut. Mauris condimentum est sed velit lacinia placerat. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nullam diam metus, pharetra vitae euismod sed, placerat ultrices eros. Aliquam tincidunt dapibus venenatis. In interdum tellus nec justo accumsan aliquam. Nulla sit amet massa augue.

*Keywords:* Needham-Schroeder Public Key Protocol, Otway-Rees Protocol, Kerberos Protocol, Model Checking

## 1. Introduction

Security protocols provision the authentication procedure of message passing among authorized users while ensuring non-transmission of information to the un-authorized agents. While the protocols are designed to ensure the same, it is equally important to find scenarios where the protocol fails. Formal analysis is one such method to detect a possible attack. The project presents the formal analysis of three protocols in an order where each protocol is an improvement over the previous. Despite this fact, each protocol has a possible attack detected.

Section 2 describe the functioning of protocols relevant to this project. Section 3 describes the components and the design that is used to model each protocol in NuXmv. Section 4 describes the implementation details of the models. Section 5 comprises the results obtained from the execution of the models and the nature of attack on the protocols that were found.

## 2. Background Details

Maecenas [**?** ] fermentum [**?** ] urna ac sapien tincidunt lobortis. Nunc feugiat faucibus varius. Ut sed purus nunc. Ut eget eros quis lectus mollis pharetra ut in tellus. Pellentesque ultricies velit sed orci pharetra et fermentum lacus imperdiet. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Suspendisse commodo ultrices mauris, condimentum hendrerit lorem condimentum et. Pellentesque urna augue, semper et rutrum ac, consequat id quam. Proin lacinia aliquet justo, ut suscipit massa commodo sit amet. Proin vehicula nibh nec mauris tempor interdum. Donec orci ante, tempor a viverra vel, volutpat sed orci.

### *2.1. Needham-Schroeder Public Key Protocol*

Needham Schroeder Public-Key Protocol aims at providing mutual authentication between two agents that communicate on a network via public keys. The two agents, $A$ and $B$ involved in the communication have public keys $K_A$ and $K_B$ that are available to the public for encryption purpose only. Both agents $A$ and $B$ are known to generate nonces, which are unique numbers used for authentication purpose during session setup by the generating agents.

Table 1: Symbols used in Needham-Schroeder Public Key Protocol

| | |
|---|---|
| $A$ | Identifier of agent $A$ |
| $B$ | Identifier of agent $B$ |
| $N_A$ | Unique random number generated by agent $A$ |
| $N_B$ | Unique random number generated by agent $B$ |
| $K_A$ | Public key of $A$ |
| $K_B$ | Public key of $B$ |

The message exchanges that take place in the protocol are as follows :

1. $A \longrightarrow B : \{N_A, A\}_{K_B}$
   $A$ sends $B$ its nonce and identity. $B$ decrypts the message and confirms the identification of the sender, $A$.
2. $B \longrightarrow A : \{N_A, N_B\}_{K_A}$
   After $B$ sends its nonce $N_B$ together with $N_A$, $A$ decrypts the message and matches its sent $N_A$ with the received $N_A$ and authenticates $B$ as an honest receiver.

3. $A \longrightarrow B : \{N_B\}_{K_B}$
   After $A$ the nonce $N_B$ to$B$, $B$ decrypts the message and matches its sent $N_B$ with the received $N_B$ and authenticates $A$ as an honest sender.

*2.2. Otway-Rees Protocol*

Otway-Rees Protocol distributes sessions keys generated by a server to the honest agents to allow safe exchange of messages between them. The session keys sent to the honest agents is always sent scrambled to avoid interception by intruders. The honest agents $A$ and $B$ request the server $S$ for their session key, where $B$ is the point of interaction with the server, $S$.

Table 2: Symbols used in Otway-Rees Protocol

| | |
|---|---|
| $A$ | Identifier of agent $A$ |
| $B$ | Identifier of agent $B$ |
| $S$ | Server's name |
| $Sn$ | Session number |
| $N_A$ | Unique random number generated by agent $A$ |
| $N_B$ | Unique random number generated by agent $B$ |
| $K_{AS}$ | Symmetric key shared between $A$ and $S$ |
| $K_{BS}$ | Symmetric key shared between $B$ and $S$ |
| $K_{AB}$ | Symmetric key shared between $A$ and $B$ |

The message exchanges that take place in the protocol are as follows :

1. $A \longrightarrow B : \{N_A, Sn, A, B\}_{K_{AS}}$
   $A$ sends relevant information to $S$ via $B$ in order to receive a session key to use for secure exchange of information with $B$ later.
2. $B \longrightarrow S : \{N_A, Sn, A, B\}_{K_{AS}}, \{N_B, Sn, A, B\}_{K_{BS}}$
   $B$ generates its own message packet and relays the received message from $A$ to server, $S$.
3. $S \longrightarrow B : \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}}$
   $S$ decrypts both message packets to verify the session number, $Sn$ in which the honest agents want to communicate and to verify their identities using $A$ and $B$. $S$ then generates the session key and sends it along with the respective nonces for verification purpose.
4. $B \longrightarrow A : \{N_A, K_{AB}\}_{K_{AS}}$
   $B$ decrypts the packet encrypted with $K_{BS}$ and accepts the session key after verifying its nonce, $N_B$. $B$ then relays the other packet to $A$ and $A$ accepts the session key after verifying its nonce, $N_A$.

*2.3. Kerberos Protocol*

The Kerberos protocol authenticates by using shared secrets. The protocol verifies the identity of the agents by confirming that both agents have the same piece of secret information (session key). The agents authenticate themselves with the server by sending their identifiers and the server sends session keys with lifetime. The lifetime of the session keys disallows a session to last forever. Timestamp plays a significant role in authenticating and determining the validity of the session key. It is used to ensure that the session key is recent.

Table 3: Symbols used in Kerberos Protocol

| | |
|---|---|
| $A$ | Identifier of agent $A$ |
| $B$ | Identifier of agent $B$ |
| $S$ | Server's name |
| $T_S$ | Local timestamp of server, $S$ |
| $T_A$ | Local timestamp of agent, $A$ |
| $L$ | Lifetime of the ticket, $K_{AB}$ |
| $K_{AS}$ | Symmetric key shared between $A$ and $S$ |
| $K_{BS}$ | Symmetric key shared between $B$ and $S$ |
| $K_{AB}$ | Symmetric key shared between $A$ and $B$, also called ticket |

The message exchanges that take place in the protocol are as follows :

1. $A \longrightarrow S : \{A, B\}$
   $A$ sends the its own identifier and the identifier of the agent with whom it wants to establish a connection. $S$ verifies the agents.
2. $S \longrightarrow A : \{K_{AB}, T_S, L, B\}_{K_{AS}}, \{K_{AB}, T_S, L, A\}_{K_{BS}}$
   $S$ generates a session keys and sends two copies of the session keys together with its local time and lifetime of the key encrypted with the respective symmetric keys of both agents.
3. $A \longrightarrow B : \{T_A, A\}_{K_{AB}}, \{K_{AB}, T_S, L, A\}_{K_{BS}}$
   $A$ decrypts its own copy, creates a packet containing its local time & identifier encrypted with the session key. $A$ also relays the other packet as it is to the the agent $B$ with whom it intends to establish a safe communication.
4. $B \longrightarrow A : \{T_A + 1\}_{K_{AB}}$
   $B$ decrypts the packet which was sent by the server originally and extracts the session key , $K_{AB}$ to use it to decrypt the other packet generated by $A$ to extract the timestamp. $B$ sends the increased timestamp

encrypted with the session key , so that $A$ could verify that $B$ indeed had the same session key as $A$.

## 3. Analysis and Design

This section describes how each protocol is modelled in NuXmv. The design includes representation of the components and abstraction of the complex features of a protocol while preserving the major functionality. Analysis refers to establishment of connection between the protocol and the design in terms of the specifications. Model validation in the protocol refers to allowance of undesirable communications between the components; however they do not lead to successful completion of the transaction.

### *3.1. Needham-Schroeder Public Key Protocol*

#### *3.1.1. Model*

The NuXmv model includes the following modules to represent the different components of Needham-Schroeder Public Key Protocol:

- `module alice` : To represent agent $A$
- `module bob` : To represent agent $B$

Each message packet traversing in the channel includes 2 fields encrypted using a certain key. The parameters to a module include two such packets one for sending, the other for receiving. Each of the parameters are allowed to hold certain values pertaining to the protocol execution. The parameters that each module take are :

- `send1`, `send2` , `sendkey`: To represent packet sent by an agent.
- `rcv1`, `rcv2` , `rcvkey` : To represent packet received by an agent.

Each agent (or module) also have a set of states that refer to the stage in which the agent is in during the execution of the protocol :

- For agent $A$ : {`a_initiate`,`a_wait`,`a_rcv_and_send`,`a_done`}
- For agent $B$ : {`b_free`,`b_receive`,`b_checks_sender`,`b_done`}

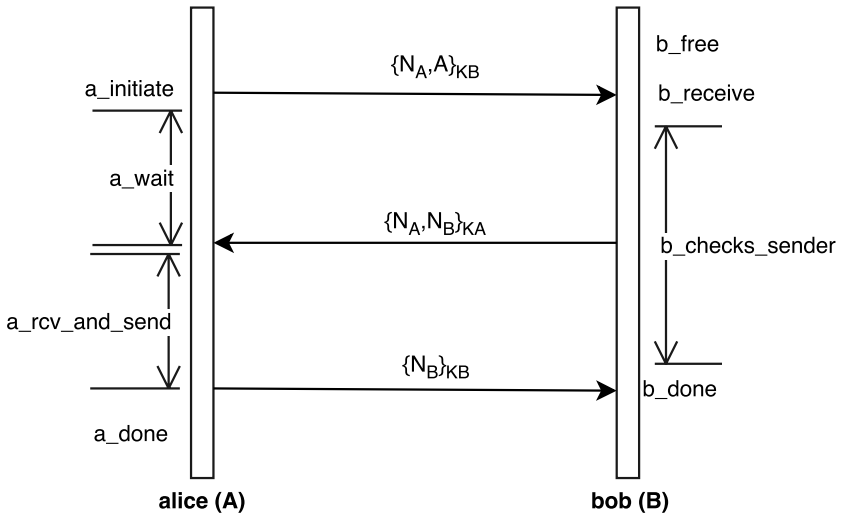


Figure 1: Model of Needham Schroeder Protocol

*3.1.2. Specifications*

- 
- 

*3.2. Otway-Rees Protocol*

*3.2.1. Model*

The NuXmv model includes the following modules to represent the different components of Otway-Rees Protocol:

- `module alice` : To represent agent $A$
- `module bob` : To represent agent $B$
- `module server` : To represent server $S$

Each message packet traversing in the channel includes 3 fields encrypted using a certain key. The 3 fields can carry nonce, identifiers, keys or can be empty depending upon the stage of the protocol execution. The parameters to a module include two such packets one for sending, the other for receiving. Each of the parameters are allowed to hold certain values pertaining to the protocol execution. The parameters that each module take are :

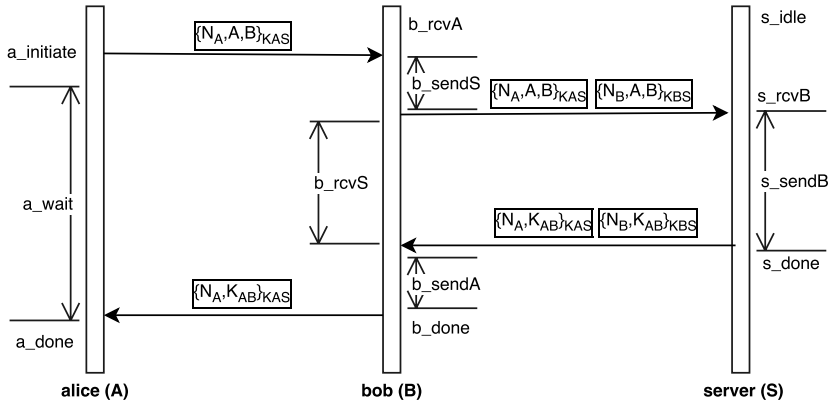


Figure 2: Model of Otway-Rees Protocol

- `send1`, `send2` , `sendkey`, `sendid`: To represent packet sent by an agent wherein `sendid` holds the value of the identifiers being sent in the message packet, `send1` holds the nonce and `send2` holds the a key or no key.
- `rcv1`, `rcv2` , `rcvkey`, `rcvid` : To represent packet received by an agent `rcvid` holds the value of the identifiers being received in the message packet, `rcv1` holds the nonce and `rcv2` holds the a key or no key.

Each module also have a set of states that refer to the stage in which the agent is in during the execution of the protocol :

- For agent $A$: {`a_initiate`,`a_wait`,`a_done`}
- For agent $B$: {`b_rcvA`,`b_sendA`,`b_rcvS`,`b_sendS`,`b_done`}
- For server $S$: {`s_idle`,`s_rcvB`,`s_sendB`,`s_done`}

*3.2.2. Specifications*

-
-

*3.3. Kerberos Protocol*

*3.3.1. Model*

The NuXmv model includes the following modules to represent the different components of Kerberos Protocol:

- `module alice` : To represent agent $A$
- `module bob` : To represent agent $B$
- `module server` : To represent server $S$

Each message packet essentially consists of 5 fields namely, encryption key, session key, timestamp, lifetime and identifier. Each agent can send or receive 2 such packets which are referred to as "chunks". 2 chunks together form a `send_msg` or `rcv_msg`. The parameters that each module take are :

- `send_msg`: To represent packet sent by an agent/server which contains 2 chunks where each chunk has 5 fields as mentioned above.
- `rcv_msg` : To represent packet received by an agent/server which contains 2 chunks where each chunk has 5 fields as mentioned above.

Each module also have a set of states that refer to the stage in which the agent is in during the execution of the protocol :

- For agent $A$: {`a_idle,a_initiate_s,a_send_s,a_rcv_s,a_send_b,a_done`}
- For agent $B$: {`b_idle,b_rcv_a,b_send_a,b_done`}
- For server $S$: {`s_idle,s_rcv_a,s_send_a,s_done`}

Each agent has a set of counters to keep track of number sessions or interactions that are active. These counters are useful in formulation of the authentication and secrecy specifications.

- `a_begincount_b` : Counter in `alice` to keep track of number of sessions initiated by `alice` with `bob`
- `a_sendSK_b` : Counter in `alice` to keep track of number of session keys sent by `alice` to `bob`

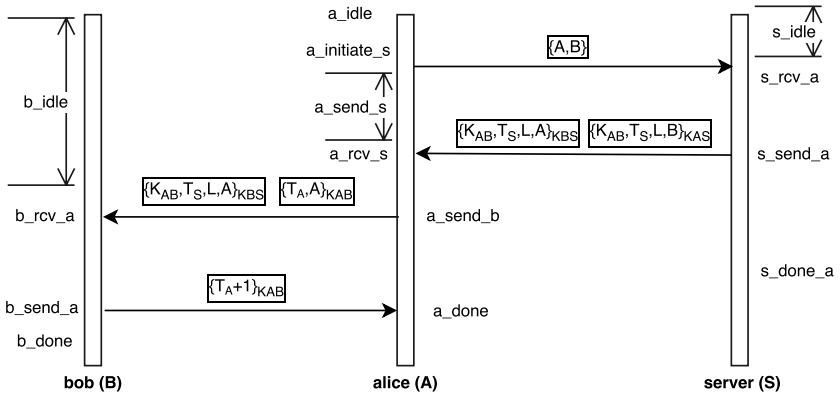


Figure 3: Model of Kerberos Protocol

- `b_endcount_a` : Counter in `alice` to keep track of number of sessions ended by `bob` with `alice`
- `b_rcvSK_a` : Counter in `alice` to keep track of number of session keys received by `bob` from `alice`

The timestamp in this model is assumed to be between 0 and 10. The lifetime is set to a constant value in order to abstract away an active session. The major assumption in the design is only beginning of a session is modelled. A running session is assumed to be a secure session and hence, not designed.

*3.3.2. Specifications*

- 
- 

## 4. Implementation

Reference to Section **??**. Etiam congue sollicitudin diam non porttitor. Etiam turpis nulla, auctor a pretium non, luctus quis ipsum. Fusce pretium gravida libero non accumsan. Donec eget augue ut nulla placerat hendrerit ac ut mi. Phasellus euismod ornare mollis. Proin tempus fringilla ultricies. Donec pretium feugiat libero quis convallis. Nam interdum ante sed magna congue eu semper tellus sagittis. Curabitur eu augue elit. Aenean eleifend purus et massa consequat facilisis. Etiam volutpat placerat dignissim. Ut

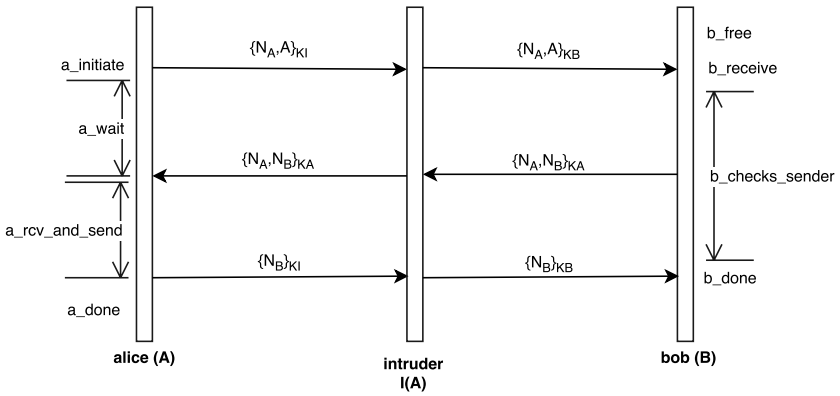


Figure 4: Attack on Needham-Shroeder Public Key Protocol

nec nibh nulla. Aliquam erat volutpat. Nam at massa velit, eu malesuada augue. Maecenas sit amet nunc mauris. Maecenas eu ligula quis turpis molestie elementum nec at est. Sed adipiscing neque ac sapien viverra sit amet vestibulum arcu rhoncus.

## 5. Test and Results

### *5.1. Needham Schroeder Public Key Protocol*

#### *5.1.1. Attack*

An attack in Needham-Schroeder Public Key Protocol suffers an attack due to intruder impersonation. If the intruder $I$ manages to establish a connection with honest agent $A$ to impersonate as $B$, denoted as $I(B)$. $I$ is able to extract the secret from the packets that are sent to it from $A$ and $A$, at different stages of the protocol, sends secrets $N_A$ and $N_B$ to $I$ thinking it to be an honest agent. The attack is represented in Figure 4.

#### *5.1.2. Violated Specifications*

- 
- 

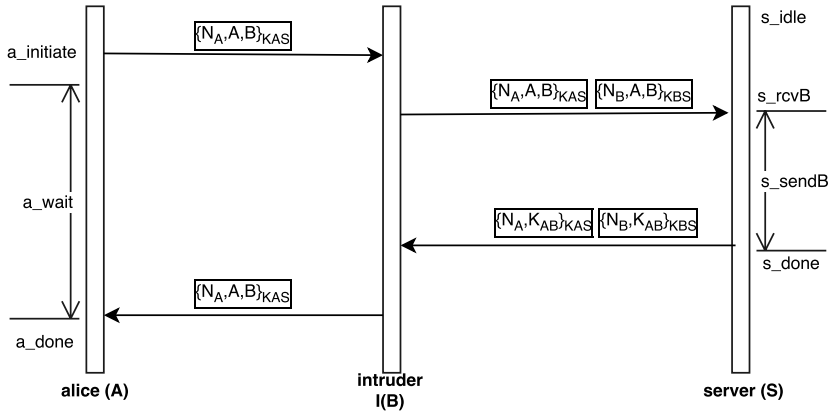


Figure 5: Attack on Otway-Rees Protocol

### *5.2. Otway-Rees Protocol*

#### *5.2.1. Attack*

An attack in Otway-Rees Protocol suffers a man-in-middle attack. In the attack, the intruder $I$ manages to impersonate as $B$, denoted as $I(B)$ and establishes a connection with $A$. In the last step of the protocol execution, $I(B)$ sends back the same packet to $A$ that it received in the first step, which now $A$ mistakes the identifiers to be the new session key. Now both $A$ and $I(B)$ have the same session key which is used for securing the subsequent message exchanges. The attack is represented in Figure 5.

#### *5.2.2. Violated Specifications*

- 
- 

### *5.3. Kerberos Protocol*

#### *5.3.1. Attack*

An attack in Kerberos Protocol is termed as replay attack. In this attack, the intruder $I(A)$ manages to intercept the message while $A$ sends the session key to $B$ and connects with $B$. Now, $B$ assumes that it has established two sessions with $A$ and it continues towards completing the protocol execution. After the protocol reaches the completion step, $B$ keeps sending the secret messages in both session, without realizing that one of the session is with a intruder. The attack is represented in Figure 6.

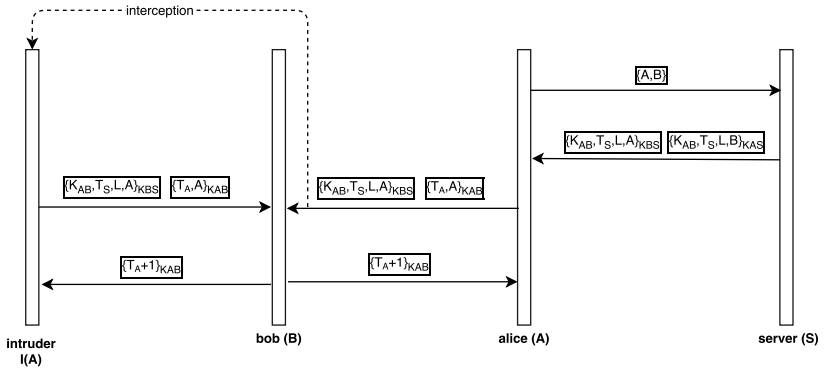


Figure 6: Attack on Kerberos Protocol

### *5.3.2. Violated Specifications*

- 
- 

## 6. Conclusion

Maecenas [**?** ] fermentum [**?** ] urna ac sapien tincidunt lobortis. Nunc feugiat faucibus varius. Ut sed purus nunc. Ut eget eros quis lectus mollis pharetra ut in tellus. Pellentesque ultricies velit sed orci pharetra et fermentum lacus imperdiet. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Suspendisse commodo ultrices mauris, condimentum hendrerit lorem condimentum et. Pellentesque urna augue, semper et rutrum ac, consequat id quam. Proin lacinia aliquet justo, ut suscipit massa commodo sit amet. Proin vehicula nibh nec mauris tempor interdum. Donec orci ante, tempor a viverra vel, volutpat sed orci.